# Chapter 2

* How to create network app

→ write Program that:

a) run of different end systems.

b) Communicate network.

c) e.g: web server SW Communicates with server SW.

* Why is no-need to write SW for network-core devices?

a) They don't run user apps.

b) apps. on end systems allows for rapid app development, Propagation.

App. Architectures

client-server

Peer-to-Pear (P2P)

1) client-Server architecture (high complex of management)

Server

→ always on-host → Permanent IP address.

→ data centers for scaling.

clients

→ Communicate with server. → may be intermittently Connected.

→ may have dynamic IP address.

→ do not Communicate directly with each other.

## 2) P2P architecture

→ No always-on server.
→ arbitary end systems directly communicate.
→ Peer request service from other peers, Provide service in return to other Peers.
→ Peers are intermittently connected and change IP address.
→ Complex management

### What is self Scalability

↳ new Peers bring new service capacity, as well as new service demands.

### Process Communicating

Process → Program running within a host
client Process → Process that initiates Communication.
server " → " " waits to be Contacted.

a) within same host → two Processes Communicate using inter-Process Communication (defined by OS)
b) within different hosts → ~~Communicate~~ Processes Communicate by exchanging messages.

* apps. with P2P arch. have (client, server) Processes

# Sockets

→ Process sends/receives messages to/from its socket

* socket analogous to door

a) sending process shoves message out door.

b) sending process relies on transport infrastructure on other side of door to deliver message to socket at receiving process.

## Addressing Processes

→ to receive messages, process must have identifier.

→ host device has unique 32 bit IP address

↳ Does it suffice for identify process?

↳ no, many processes can run on same host.

identifier → includes IP address and port numbers.

## Types of messages

a) message syntax

↳ what fields in messages & how fields are delinated

b) message semantics

↳ meaning of information in fields.

## *non-Persistent HTTP

a) at most one object sent over TCP, connection then closed.

b) download multiple objects required multiple connections.

## *Persistent HTTP

→multiple objects can be sent over single TCP connection between client, server.

## Uploading from input

### a) POST method

→web page often includes from input

→input is uploaded to server in entity body.

### b) URL method

→uses GET method.

→inputs is uploaded in URL field of request line

HTTP/1.0 :

→GET

→POST

→Head

HTTP/1.1 :

* GET, POST, Head

* PUT

* Delete

# HTTP response status Code

→ status code appears in 1st line in server-to-client response message.

* examples

* 200 ok → request succeeded, requested object later in this msg.

* 301 moved Permanently → request object moved, new location specified later in this message.

* 400 Bad request → request msg. not understood by server

## Four Components of user-server state: Cookies

1) Cookie header line of HTTP response msg.

2) " " " in next request msg.

3) " kept on user's host managed by user's browser

4) back-end database at web site.

* when initial HTTP requests arrives at site, site creates
→ unique ID.
→ entry in back end database for ID.

## * uses of Cookies

a) authorization  b) shopping carts

c) recommendations.  d) user session state (web e-mail)

## How to Keep "state"

→ Protocol endpoints: maintain state at sender/receiver over multiple transactions.

→ cookies: http messages carry state.

### Notes

→ Cookies permit sites to learn alot about you.

→ you may apply name & e-mail to sites.

---

## Web caches (Proxy server)

**Goal** satisfy client request without involving origin server

**Why web caching**

Caching example (2.3g slides)

a) reduce response time for client request.

b) reduce traffic on institution's access link.



proxy server: server that stores requested objects.

## Conditional Get

Goal don't send object if cache has up-to-date cached version.

→ no object transmission delay.
→ lower link utilization.

Cache specify data of cached copy in HTTP request. if-modified-since: <data>

Server response contains no object if cached copy is up-to-date. (HTTP/1.0 304 NOT ~~Found~~ modified)

## FTP (file transfer protocol)

→ transfer file to/from remote host.
→ client/server model
client: initiates transfer.
server: remote host.

## FTP Commands

→ sent as ASCII text over control channel
* USER → username, PASS → password.
* LIST → return list of file in current directory.

* RETR filename → retrieves (gets) file
* STOR filename ⇒ stores (puts) file onto remote host

## Electronic mail

### Components (major)

* user agents
* mail servers.
* simple mail transfer protocol: SMTP

### user Agent

→ composing, reading, editing mail messages.
→ e.g: outlook, iPhone mail client.
→ outgoing, incoming msgs stored on server.

### mail Servers

* mail box contains incoming msgs for user.
* message queue of outgoing mail msgs.

### SMTP Protocol

↳ between mail servers to send mail msgs.

client : sending mail server.

server : receiving " "

# SMTP (RFC 2821)

→ uses TCP to transfer email msg from client to server.

→ direct transfer → sending server to receiving server.

* Three phases of transfer:

a) handshaking. b) transfer of msgs.

c) closure.

Command/response interaction

Commands ——→ ASCII text

response → status code and phrase.

| Compare between HTTP & SMTP | |
|---|---|
| HTTP | SMTP |
| pull | push |
| each object encapsulated in its own response msg | multiple objects sent in multipart msg. |

## *PoP (Post office Protocol)

↳ download and keep copies of msgs on different clients.

↳ statless across sessions. } disadvantages

↳ client oriented

## IMAP (internet mail Access Protocol)

→ Keep all msgs in one Place (server)

→ allow user to organize msgs in folders.

→ Keep user state across sessions

---

## DNS (domain name system)

a) distributed data base implemented in hierarchy of many name servers.

b) application-layer Protocol: hosts name servers communicate to resolve names.

* why not Centralize DNS?

a) single Point of failure. b) traffic volume

c) distant Centralized database. d) maintenance.

---

if client wants IP for www.amazon.com

* client queries root server to find .com DNS server.

* client " .com DNS server to get amazon.com DNS server.

* client queries amazon.com DNS server to get IP address for www.amazon.com

→ look at last Page

## Root name servers

a) Contacted by local name server that can't resolve name.

b) Contacts authoritative name server if name mapping not known.

c) gets mapping.

d) return mapping to local name server.

## Top-level domain (TLD) server.

→ responsible for com, org, edu, jobs, .....

→ network solutions maintains servers for .com TLD.

→ educause for .edu TLD.

## authoritative DNS servers

→ organization's own DNS server

→ Provide authoritative hostname to IP mapping for organization's named hosts.

→ Can be maintained by organization or service provider.

## Local DNS name server

- → does not strictly belong to hierarchy.
- → each ISP has one
- → also called "default name server"
- → when host makes DNS query, query is sent to its local DNS server.

### Notes

- → one name server learns mapping, it caches mapping.
- → cache entries timeout (disapear) after some time (TTL)
- → TLD servers typically cached in local name servers.

## DNS records

DNS: distributed db storing resource records (RR)

RR format: (name, value, type, ttl)

### type A

* name is hostname.
* value is IP address

### type = NS

* name is domain (foo.com)
* value is hostname of authoritative name server for this domain

### type = MX

↳ value is name of mailserver associated with name.

- type = CNAME

→ name is alias name for some "Canonical" name.

→ value is Canonical name

## DNS Protocol, messages

→ query and reply msgs, both with same msg format.

msg header (identifier, flags)

* identifier 16 bit # for query, reply to query uses same #.

* flags

* query or reply.
* recursion available.
* recursion desired.
* reply is authoritative.

* questions (name, type fields for a query)

* answers (RRs in response to query)

* authority (records for authoritative servers)

* additional info (additional "helpful" info that may be used)

# Inserting records into DNS

* example : new startup "Network Utopia"

* register name networkutopia.com at DNS register.
  - Provide names, IP address of authoritative name server.
  - register inserts two RRs into .com TLD server.

* Create authoritative server type A record for www.networkutopia.com ; type MX record for networkutopia.com.

File ← جزء زائد في الـ

## Advantages of Cookies

1) Save Processing by not needing to authenticate every time.
2) Save user customizations and Configuration for a website (if available

## DNS

### amazon.com

1) Root server redirct to .com TLD servers.
2) TLD servers redirect to amazon servers.
3) amazon servers then send list of servers.